How to Install OCTA8.4 on macOS

July 27, 2022 J.Takimoto

CONTENTS:

[0] Overview

OCTA consists of two main components; Gourmet and simulation engines.
All engines use UDF files as their input/output. Gourmet is a GUI application to edit/analyze UDF files.

Gourmet:
You need to build Gourmet by yourself, but it should be easy; see [1] and [2].
If you have any trouble, feel free to ask in OCTA BBS.

Simulation engines (Cognac, Pasta, Sushi, Muffin, Kapsel):
Pre-built binaries of these engines are included and should work on your Mac; see [3].
If they do not work, you can easily re-build them; see [5].

NOTE: If you are upgrading from OCTA83, and if there is a line like

```
source /path/to/OCTA83/GOURMET/gourmetterm -
```

in your shell startup file (`.bashrc` or `.zshrc`), then remove the line and restart the shell.

[1] Required software for Gourmet

Please install the following software (1.1) ~ (1.4) if not yet installed.
(If you are upgrading from OCTA83 you already have (1.1), (1.3) and (1.4).)
gnuplot is required only when creating a plot by Gourmet. You can build Gourmet before installing gnuplot.

(1.1) Apple's "Command Line Tools"

Open Terminal.app, and type (`%` is the shell's prompt):

```
% xcode-select --install
```

(1.2) OpenJDK 11

For arm64 (Apple silicon), Azul Zulu Builds is the only OpenJDK available now (July 2022):
https://www.azul.com/downloads/

For Intel Mac, we are mainly testing with AdoptOpenJDK, but other OpenJDK should also work:
https://adoptopenjdk.net

If you already have JDK 8 (for example for OCTA83), you need not uninstall it, but you need to install OpenJDK 11 also.
(if you have Oracle's JDK 8 then better to remove it unless you have a valid license).

(1.3) Python3 (3.6 ~ 3.10) with numpy

IMPORTANT: You can NOT use `/usr/bin/python` (=python2.7) or `/usr/bin/python3` even if they exist. You need to install Python3 for Gourmet by yourself.

(step-1) Install Python3 (3.6 ~ 3.10) from one of the below (or other sources you like):

- ○ "Official" Python from python.org:
  https://www.python.org/downloads/
- ○ Package managers such as Homebrew, MacPorts, etc.
- ○ pyenv (only for experienced users):
  https://github.com/pyenv/pyenv
- ○ Miniconda (only if you are sure that you have a valid license):
  https://docs.conda.io/en/latest/miniconda.html

Any of these would work, although only the official Python is well-tested.

(step-2) After installing the Python, start a *new* shell (open a *new* Terminal window), and:

```
% which python3
/path/to/python3          # is this the one you want to use? (it should not be /usr/bin/python3)
```

If the output corresponds to the `python3` you have installed for Gourmet, then go to **(step-3)** below. For example, if you have installed the Official Python (from python.org), the output should be either

```
  /usr/local/bin/python3
```

or

```
  /Library/Frameworks/Python.framework/Versions/3.x/bin/python3
```

(you can use either of them interchangeably). If you are using Homebrew etc. the output will differ, but it should NOT be `/usr/bin/python3`.

IMPORTANT: If the output of "`which  python3`" does not correspond to the `python3` you have installed for Gourmet, your `python3` is not at the top of your `PATH`. Probably you need to do "something" for setting `PATH` properly. Please consult the install document of the Python3 you have chosen. Or, if you are sure that you know the full pathname of the `python3` command you want to use, then use the full pathname when installing numpy in (step-3) below, and also when building Gourmet in [2] (2.1).

(step-3) install numpy:

- ○ If you are using Official Python:

```
% python3 -m pip install numpy
```

  or (if you want to install only for you)

```
% python3 -m pip install --user numpy
```

- ○ If you are using Homebrew etc., you may be able to install numpy by the package manager.
  If it is not supported by the package manager, you may try using pip as above.

- ○ If you are using Minoconda (make sure that you have a valid license):

```
% conda install numpy
```

(1.4) Gnuplot

You can install gnuplot by one of the following methods:

(a) If you are using Homebrew etc., you can install gnuplot by using the package manager.
If you are using Apple silicon (arm64), this may be the only easy way to install gnuplot.

(b) Prebuilt binary of gnuplot for macOS may (or may not) be found at:
http://ricardo.ecn.wfu.edu/pub/gnuplot/

(c) If you are using intel Mac (x86_64) you can easily build/install gnuplot by yourself:

1. Download AquaTerm from
   https://sourceforge.net/projects/aquaterm/
   and install it. It is a standard macOS installer.

2. Download the latest source code of gnuplot (gnuplot-x.y.z.tar.gz) from
   https://sourceforge.net/projects/gnuplot/files/gnuplot/

3. In Terminal.app:

```
% tar xvf gnuplot-x.y.z.tar.gz
% cd gnuplot-x.y.z
% export CPPFLAGS='-F/Library/Frameworks'
% export LDFLAGS='-F/Library/Frameworks'
% ./configure --with-aquaterm --with-readline=builtin
% make
% sudo make install
```

   This will install gnuplot into `/usr/local/bin`.

After installing gnuplot, please make sure that it is in your `PATH`.

NOTE: gnuplot is required only when you use Gourmet (and create a plot). You can build Gourmet before installing gnuplot.

NOTE: (After you have built Gourmet): If you create a plot from within Gourmet but the plot disappears immediately, then probably gnuplot is using the terminal "`wxt`". If this happens, please add the following line

`set term aqua`   (only available on intel Mac)

or

`set term qt`

to the file `.gnuplot` in your home directory (create the file if it does not exist).
At least one of the above should work; please experiment after Gourmet has been built.

[2] How to build and run Gourmet

Once the required software (1.1)-(1.3) is installed, building Gourmet should be easy.

(2.1) Build Gourmet

```
% cd /path/to/OCTA84/GOURMET/src
% ./build-gourmet
We will use the following python3:
/path/to/python3
 OK? [y/N]:
```

If `/path/to/python3` is the python3 you want to use for Gourmet, answer **y**; then the build will start (it will take a few to 10 minutes).

If it is not the `python3` you want to use, either fix your `PATH`, or (if you know the correct full pathname of your python3)

```
% ./build-gourmet /path/to/your/python3
```

where `/path/to/your/python3` is the full pathname of the python3 you want to use for Gourmet.

(2.2) Start Gourmet from command prompt

You can start Gourmet by the following shell script:

```
% /path/to/OCTA84/GOURMET/gourmet
```

You may add `/path/to/OCTA84/GOURMET/` to your `PATH`, or create a symlink to the script in a directory in your `PATH`. For example, if `~/bin` is in your `PATH` (~ is your home directory)

```
% cd ~/bin
% ln -s /path/to/OCTA84/GOURMET/gourmet .
```

Then you can start Gourmet by

```
% gourmet
```

or

```
% gourmet xxx.udf [yyy.udf ...]
```

It will create a new Terminal window (minimized in the Doc) and start Gourmet in that Terminal.

(2.3) StartGourmet.app (only on Catalina or earlier macOS)

An application StartGourmet.app is created in `OCTA84/GOURMET/src/osxapp/` (it will not run on Big Sur due to the Apple's security restrictions). You can add this app to the Dock so that you can start Gourmet by clicking the Dock icon. You can also drag & drop a UDF file on it (you need *not* move the app to other places such as `/Applications`).

Or create an alias of StartGourmet.app on Desktop (or anywhere you like).

Or you can set (in Finder) StartGourmet.app as the default application to open UDF files. Then you can double-click a UDF file to open it by Gourmet.

[3] How to run simulation engines

(3.1) Set environment variables

Prebuilt binaries of the simulation engines are in `OCTA84/ENGINES/bin/macosx/`.
(if these binaries do not work, you need to rebuild them; see [4]). HDF5 tools (such as h5dump, used for analyzing kapsel output files) are in `OCTA84/ENGINES/kapsel4.10/hdf5/bin/`.

Before running simulation engines, you need to set environment variables used by the engines, and add /*path*/*to*/`OCTA84/ENGINES/bin/macosx/` to your `PATH`.
This is done by:

```
% source /path/to/OCTA84/GOURMET/engine_env
```

You need to do this for each shell session. You may add the '`source ...`' to your shell startup file if you want it to be sourced in every shell session.

(3.2) Run the engines

If you want to run (for example) Cognac:

```
% cognac101 -I input.udf -O output.udf
```

IMPORTANT: see (3.3) below before running Kapsel.

(3.3) Multithreading

Cognac and Kapsel support multithreading with OpenMP.

○ Cognac
You can specify the number of threads by the option `-n`:

```
% cognac101 -n 4 -I input.udf -O output.udf
```

If `-n` is not given Cognac will use only single thread.

○ Kapsel
You *should* specify the number of threads by the environment variable `OMP_NUM_THREADS`

```
% export OMP_NUM_THREADS=4
% kapsel -Iinput.df -Ooutput.udf -Ddefine.udf -Rrestart.udf
```

If `OMP_NUM_THREAD` is not set, default is the number of *logical* cores (including Hyper-Threading), which is not optimal (and may cause problems in some cases).

[4] Using UDFManager in Python

There are several python modules (UDFManager, CognacBasicAnalysis, etc.) for the analysis of UDF files. You need to set PYTHONPATH correctly to use them. This can also be done by sourcing engine_env:

```
% source /path/to/OCTA84/GOURMET/engine_env
% python3
>>> import UDFManager from UDFManager
```

See Chapter3 of
    DOCUMENTS/{english,japanese}/gourmet/PythonInterface_{eng,jpn}.pdf
and Section 7.4 of
    DOCUMENTS/{english,japanese}/engine/Cognac_{eng,jpn}.pdf).

UDFManager depends on a binary module UdfManagyerPython.so in OCTA84/GOURMET/lib/macosx, If it doesn't work on your Mac you can easily rebuild it by:

```
% cd /path/to/OCTA84/GOURMET/src
% ./build-pymod /path/to/your/python3
```

[5] How to rebuild simulation engines

You can rebuild all the engines by:

```
% cd /path/to/OCTA84/ENGINES
% ./build-engines
```

Or you can rebuild each engine in its source directory. For example (modify '-j 4' appropriately):

```
% cd /path/to/OCTA84/ENGINES/COGNAC1014/src
% source /path/to/OCTA84/GOURMET/engine_env
% make -j 4                    # run 4 parallel jobs for the build
% make install
```

If the build fails, please rebuild libplatform.a (UDF I/O library used by engines) by:

```
% cd /path/to/OCTA84/GOURMET/src
% ./build-lib
```

and try rebuilding the engines again.

Note for rebuilding Kapsel:

Kapsel uses an external library HDF5. Binary of this library is included in OCTA84/ENGINES/kapsel4.10/hdf5/lib/. If you want to rebuild both this library and kapsel, please use the script OCTA84/ENGINES/kapsel4.10/build-kapsel.

[6] Known problems

(1) In Gourmet's Editor window, you need to double-click on a folder-like icon to open the data structure; clicking on a triangle icon does not work (only in the Tree View).
(2) If you start "picking" (ctrl-click) on a Viewer window, you need to hit ESC key to end the picking mode; just releasing the ctrl-key is not enough.
(3) If you are using a Retina display, the Viewer window may have some minor problems (e.g., too small fonts in legends).

[7] How to uninstall OCTA

1. Remove the directory `OCTA84`
2. If you have copied (or created a symlink to) the script '`gourmet`' in the step (2.2) above, remove it.
3. Remove the directory `~/.gourmet/` (if it exists).

[A1] Additional components

Two additional components **ImageLoader** and **AITool** are included in this release.

Both are *experimental* on macOS and may not work as expected.
See tutorials and manuals in `DOCUMENTS/{english,japanese}/gourmet/` for what can be done by these tools.

ImageLoader:

See `GOURMET/tool/ImageLoader/README_mac.txt` for how to install ImageLoader.

AITool:

AITool is not supported on arm64 (Apple silicon).

See `AITool/config/Install-AITool.pdf` for how to install AITool.

You *should* be familiar with how to manage multiple versions of Python and how to create virtual environment. Otherwise, please do not try to install AITool on your Mac.

AITool requires Python3.7 and many Python modules (with specific versions). Some of them may not be available anymore when you try to install AITool.